



**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

APPEAL NO:

In Re Application of:

Huras, Matthew A. et al.

Serial No: 09/774,202

Filed: January 29, 2001

For: ONLINE DATABASE TABLE REORGANIZATION

**RECEIVED**

**AUG 04 2004**

**Technology Center 2100**

**APPELLANT'S BRIEF**

Joseph A. Sawyer, Jr.  
Sawyer Law Group LLP  
2465 E. Bayshore Road, Suite 406  
Palo Alto, CA 94303  
Attorney for Appellant(s)  
International Business Machines Corp.



## TOPICAL INDEX

I.	Real Party in Interest . . . . .	1
II.	Related Appeals and Interferences . . . . .	1
III.	Status of Claims . . . . .	2
IV.	Status of Amendment . . . . .	2
V.	Summary of the Invention . . . . .	3
VI.	Issues . . . . .	3
VII.	Grouping of Claims . . . . .	4
VIII.	Arguments . . . . .	4
	A.    Rejections Under 35 U.S.C. 102 . . . . .	4
	D.    Rejections Under 35 U.S.C. 103 . . . . .	10
IX.	Appendix . . . . .	i

**RECEIVED**  
AUG 04 2004  
Technology Center 2100



120  
AP

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In Re Application of:

Date: May 3, 2004

Matthew A. HURAS, et al.

Confirmation No: 7317

Serial No: 09/774,202

Group Art Unit: 2172

Filed: January 29, 2001

Examiner: Alam, Shahid Al

For: ONLINE DATABASE TABLE REORGANIZATION

Mail Stop Appeal Brief-Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

**RECEIVED**

AUG 04 2004

Technology Center 2100

**APPELLANT'S BRIEF ON APPEAL**

Sir:

Appellant herein files an Appeal Brief drafted in accordance with the provisions of 37 C.F.R. § 1.192(c) as follows:

**I. REAL PARTY IN INTEREST**

Appellant respectfully submits that the above-captioned application is assigned, in its entirety to International Business Machines Corporation, of Armonk, New York.

**II. RELATED APPEALS AND INTERFERENCES**

Appellant states that, upon information and belief, he is not aware of any co-pending appeal or interference which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

### **III. STATUS OF CLAIMS**

Claims 1-59 are pending. -Claims 1, 18, 30-34, 38, 44-47, 51, and 57-59 stand rejected, and claims 2-17, 19-29, 35-37, 39-43, 48-50, and 52-56 are objected to as being dependent on rejected base claims. The present application was originally filed with claims 1-59. In an Amendment dated September 19, 2003, claims 1, 18, 30, 34, 38, 44, 47, 51, and 57 were amended. Claims 1-59 are on appeal and all applied rejections concerning claims 1-59 are herein being appealed.

Appellant requests that the Amendment After Final submitted on February 3, 2004, be entered to put the claims 30 and 44 in better form for this appeal. The claims referenced herein are in the form as provided in the Amendment After Final.

### **IV. STATUS OF AMENDMENT**

Appellant filed an amendment subsequent to final rejection on February 3, 2004, in which remarks and minor amendments were presented. The Examiner responded with an Advisory Action mailed on February 10, 2004, in which the Examiner deemed that the remarks were unpersuasive and that the amendment after final did not place the application in condition for allowance, and stated reasons similar to those given in the Final Office Action.

In Appellant's original Appeal Brief filed May 6, 2004, the claims in the appendix were presented to include the amendments of the amendment after final. The Examiner mailed a Notification of Non-Compliance on June 28, 2004, indicating that the Examiner did not in fact enter the amendments after final in the application, and stating that the Appeal Brief did not correctly present headings in its argument section. In response to the Notification of Non-Compliance, Appellant has filed this modified Appeal Brief.

## **V. SUMMARY OF THE INVENTION**

The present invention provides a method and system for online reorganization of a database table. The online reorganization allows a database table to be scanned, accessed, and updated during reorganization (Specification, e.g., page 1, lines 13-15, 19-22; page 12, lines 15-19; page 13, lines 14-16; page 15, line 13 to page 16, line 17; page 20, lines 5-7; page 26, lines 5-9). In the reorganization of the database table, data records are moved from move pages in the table to available space within the same database table, and the database table can be accessed, scanned, and updated during these reorganization steps in “online” fashion, i.e., these accesses and modifications to the database table can be made while it is being reorganized. In some aspects of the invention, move steps are followed by clean up steps to remove temporary pointers provided in the move steps (Specification, page 18, line 6 to page 19, line 11). Some aspects of the invention provide constraints to scanner processes and synchronization of move steps and clean up steps with scanner processes (Specification, page 21, lines 1-15; pages 22-24).

Previous reorganization systems and methods did not allow a user online access to the database table while it was being reorganized to allow access, scans, and updates to the database table. Appellant’s invention has the advantage of allowing the user and processes to perform desired functions or operations with the database table without having to wait for the database to be finished with the reorganization process.

## **VI. ISSUES**

The issues presented are:

1. Whether Claims 30, 31, 44, 45, 57, and 58 are anticipated by Beier et al. (U.S. Patent No. 5,933,820) (hereinafter “Beier”) under 35 U.S.C. § 102(a).
2. Whether Claims 1, 18, 32-34, 38, 46, 47, 51, and 59 are unpatentable over Beier

under 35 USC § 103(a).

## VII. GROUPING OF CLAIMS

Appellant hereby states that claims 30, 31, 44, 45, 57, and 58 form a first group, and claims 1, 18, 32-34, 38, 46, 47, 51, and 59 form a second group. Thereby the claims form two independent groups.

## VIII. ARGUMENTS

### A. Rejections under 35 U.S.C. 102

#### 1. Summary of Rejections

The Office Action dated June 19, 2003 (hereinafter “Office Action”) rejected Claims 30, 31, 44, 45, 57 and 58 under 35 USC § 102(a) as being anticipated by Beier.

With regard to claim 30, the Examiner stated:

**Beier teaches vacate move step to move data records from move pages in the table to available space in the database table (column 5, lines 1-6); and**

**A fill move step to move data records into move pages in the table, wherein the database table can be accessed, scanned, and updated during the vacate move step and fill step of the reorganization (column 5, lines 1-6).**

With regard to claim 31, the Examiner stated that each move step comprises the step of defining temporary pointers from the original position of each moved record to the moved position of the moved record (col. 5, lines 1-6).

With regard to claim 44, the Examiner stated the same rejection as provided above for claim 30.

With regard to claim 45, the Examiner stated the same rejection as provided above for claim 31.

With regard to claim 57, the Examiner stated the same rejection as provided above for

claim 30.

With regard to claim 58, the Examiner stated the same rejection as provided above for claim 31.

For the reasons presented below, Appellant respectfully requests that the Board reverse the Examiner's final rejection of Claims 30, 31, 44, 45, 57, and 58.

## 2. Description of Prior Art: Beier et al. (U.S. Patent No. 5,933,820)

Beier discloses a database management system in which direct and indirect pointers are used to locate targeted data elements in a database. One or more data elements of a data set are moved to a new data set or partition, such that the old data set is recreated as the new data set and the old data set is abandoned when the reorganization is complete. After reorganization, direct pointers to data elements that have moved are not all updated at once, but are updated upon the first reference to each moved data element. A direct pointer has assigned to it a context and reorganization number that indicates the reorganization level of the target at the time the direct pointer is assigned. Upon reference to the targeted data element, the pointer's reorganization number is compared with a reorganization number assigned to the entire data partition to check whether the pointer is valid or out-of-date; if out of date, an indirect index and indirect pointer is used to reference the target data element and the direct pointer is updated so that the direct pointer may be used in later references until another reorganization involves the target data element.

## 3. The Beier Reference Does Not Teach Appellant's Invention of Claim 57

Unpatentability by anticipation under 35 U.S.C. 102(a) requires that the invention be known or used by others in this country, or patented or described in a printed publication in this

or a foreign country, before the invention thereof by the appellant for patent. Independent claim 57, argued here as an example of the independent claims of this group, recites a computer system for reorganizing a database table online. In particular, claim 57 recites, in pertinent part:

57. A computer system for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the computer system comprising:

- a) means for carrying out a vacate move step to move data records from move pages in the table to available space in the database table; and
  - b) means for carrying out a fill move step to move data records into the move pages in the table,
- wherein the database table can be accessed, scanned, and updated during the vacate move step and fill move step of the reorganization.

Claim 57 is patentable over Beier since the features of claim 57 are not disclosed or suggested by this reference. In particular, Beier fails to teach or suggest the recited features of: 1) move steps of a reorganization that allow a database table to be accessed, scanned, and updated during the vacate move step and fill move step of the reorganization; and 2) a vacate move step to move data records from move pages to available space in the database table and a fill move step to move data records into the move pages in the table.

With regard to the feature of online reorganization allowing a database table to be accessed, scanned, and updated during the reorganization, Beier does not teach or suggest this feature. In appellant's claim, the data records of the database table can be accessed, scanned, and updated during the reorganization, i.e., while move steps of the reorganization are occurring. Beier does not disclose or suggest providing a reorganization processing allowing accessing, scanning, and updating of the database table. Beier does not address, for example, scanning operations and updates during reorganization of the partition being reorganized because the Beier method either makes the data set being reorganized unavailable to scanners, as indicated in col. 7, lines 26-30 and col. 11, lines 42-45; or Beier must synchronize the incrementing of a "reorganization



number” with the absence of scanners. Nowhere in Beier is a scanning or update of records suggested prior to, after, and across move steps of a reorganization process.

The Examiner states in the Final Office Action (page 2, point 3) that Beier teaches at col. 3, lines 37-60 a multi-step reorganization process, where the steps include running a pre-reorganization utility that examines structures and determines what has to be done, running a scan utility for data that is not being reorganized but is related to data that is being reorganized, including a reload operation and also running a prefix resolution utility and a prefix update utility. The Examiner states that Beier teaches “scan utilities” that are run on one or more related databases which are being pointed into or from the database being reorganized.

However, these described “scan utilities” of Beier do not read on, and are not the same as, the scanning of the database table during reorganization as recited in claim 44. Beier specifically states that the scan utility is run for data that is not being reorganized. These scan utilities are used to determine which other data elements are referenced by the data being reorganized. This is in direct contrast to claim 57, which recites that a database table is being reorganized, and that the database table itself, i.e. data that is being reorganized, can be scanned during the move steps of the reorganization. The cited lines of col. 3 in Beier do not disclose or suggest such scanning of database data that is being reorganized.

The Examiner states in the Final Office Action (page 3) that Beier teaches at col. 5, lines 2-9 that when a database is being organized that has alternate, i.e., secondary, indexes associated with a data element being moved, at the time the data element is being moved from the old location to the new location, there is an ability to update all of the indexes.

However, this described “updating” of indexes in Beier does not read on, and is not the same as, the “updating” of the database table during reorganization as recited in claim 44. Beier specifically states in the cited lines that secondary indexes, associated with data elements being

moved, are being updated. Such indexes are pointers to the data elements being reorganized, and the update is the changing of the pointers to reflect the new location of the data elements. Updating these indexes are not the same as updating a database table—a database table is the actual data records in the database, i.e. the data elements. Appellant’s claim 57 thus recites that the data records themselves can be updated during reorganization, e.g., adding data records to the database table, deleting data records from the table, or modifying data records, by users or applications, in an online fashion, as indicated in Appellant’s specification throughout (e.g., page 1, lines 13-15, 19-22; page 12, lines 15-19; page 13, lines 14-16; page 20, lines 5-7; page 26, lines 5-9). This is not the same as updating the indexes described by Beier. There is no teaching in Beier for updating data records of a database table during move steps of the reorganization as recited by Appellant. Similarly, the “prefix update utility” mentioned by Beier at col. 3, line 46 refers to a prefix, which is a group of pointers, not data records of a database table.

In the Final Office Action, the Examiner states that a prior art disclosure need not be express in order to anticipate, and that even if a prior art inventor does not recognize a function of his or her process, the process can anticipate if that function was inherent or necessarily present as made clear by extrinsic evidence. However, there is no indication that Beier’s invention would inherently allow features of Appellant’s claim, such as, for example, updates made to a database table while that table is being reorganized, as recited by Appellant. Such a feature is not inherent to Beier, since it is described and suggested nowhere in Beier as discussed above, and further, Beier states that during reorganization, data elements of a data set are copied or “recreated” in a different portion or new data set (col. 15, lines 15-20, 24-27), which indicates that the data is being reorganized into a new database structure and storage. This type of reorganization does not allow Appellant’s online scans or updates of data in a database table during reorganization, as further indicated in Beier at col. 11, lines 42-45, which states that the rest of the database “is available for use,” while the

reorganized portion is not available for use; and at col. 4, lines 3-7. This is because an update of Beier's database table during reorganization would not be assured of correctly finding/updating recently moved or reorganized data elements. There is thus no indication of the features of Appellant's claims being inherent to Beier. Furthermore, the Examiner has cited no extrinsic evidence that shows the above-described features of claim 57 to be necessarily present in Beier's disclosure.

With regard to the second feature, means for carrying out the vacate move step to move data records from move pages to available space in the database table and the fill move step to move data records into the move pages in the table, Beier does not disclose or suggest these elements. The Examiner stated that Beier teaches a vacate move step to move data records from move pages in the table to available space in the database table, and a fill move step, at col. 5, lines 1-6. These cited lines mention only that a database element is being moved from an old location to a new location. As described above, Beier describes his process in which, during a reorganization, data elements of a data set are copied or "recreated" in a different partition or new data set (col. 15, lines 15-20, 25-27). Beier's method of copying data elements from an old set to a new, reorganized data set is not the same as moving elements from and to the same stored database table, as recited by Appellant's means for carrying out the vacate move step of claim 57. Similarly, Appellant's fill move step moves data records into the move pages in the same stored table; Beier's method, however, moves data elements to a new data set, not into the same data set from which data records were originally stored. Furthermore, Beier does not mention the combination of vacate and fill as in claim 57, where data is moved out of locations in the database table in the vacate step, and other data from the database table is then moved into the vacated locations in a fill step. Beier mentions only the generic moving of records from an old location to a new location at the cited lines.

Consequently, Beier cannot teach or suggest the method recited in claim 57.

#### 4. Claims 30, 31, 44, 45, and 58

As to claim 30, Beier does not disclose or suggest an online database table that can be scanned, accessed and updated during move steps, nor a vacate move step to move data records from move pages in the table to available space in other pages in the same stored table, nor a fill step to move data records into the move pages. Claim 30 is therefore patentable over Beier for reasons similar to claim 57.

As to claim 31, this claim is dependent on claim 30 and is patentable over Beier for at least the same reasons as claim 30, and for additional reasons.

As to claim 44, Beier does not disclose or suggest online scanning, accessing and updating during reorganization, nor a vacate move step to move data records from move pages in the table to available space in other pages in the same stored table, nor a fill step to move data record into the move pages. Claim 44 is therefore patentable over Beier for reasons similar to claims 30 and 57.

As to claim 45, this claim is dependent on claim 44 and is patentable over Beier for at least the same reasons as claim 44, and for additional reasons.

As to claim 58, this claim is dependent on claim 57 and is patentable over Beier for at least the same reasons as claim 57, and for additional reasons.

Consequently, Beier cannot teach or suggest the subject matter recited in claims 30, 31, 44, 45, and 58.

Accordingly, Appellant respectfully requests that the Board reverse the final rejection of Claims 30, 31, 44, 45, 57, and 58.

## **B. Rejections under 35 U.S.C. 103**

### 1. Summary of the Applied Rejections

The Office Action dated June 19, 2003 rejected 1, 18, 32-34, 38, 46, 47, 51, and 59 under 35 USC § 103(a) as being unpatentable over Beier. These Office Action rejections were

maintained in the Final Office Action and in the Advisory Action.

With regard to claim 1, the Examiner stated:

**Beier teaches moving a subset of records within the database table (column 5, lines 1-6); flagging each moved record as a reorganization record (column 6, lines 55-60); creating a reorganization pointer record for each moved record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record (abstract).**

**Beier does not explicitly disclose establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the organization; however, Beier includes using direct pointers result in a multi-step reorganization process ... a prefix update utility, column 3, lines 37-46 and see also column 15, lines 37-50).**

**It would have been obvious to a person of ordinary skill in the art at the time of the invention to incorporate a scanner process to Beier teaching in order to improve data availability.**

With regard to claim 18, the Examiner stated:

**Beier teaches a vacate move step (column 5, lines 1-6); a vacate clean up step (column 1, lines 36-39); a fill move step (column 5, lines 2-6).**

**Beier does not explicitly disclose a fill clean up step wherein each clean up step is synchronized to commence at the completion of a move step and to commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronized to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed; however, Beier includes using direct pointers result in a multi-step reorganization process... a prefix update utility... the work files are combined into a sort. See column 3, lines 37-63 and see also column 15, lines 37-50).**

**It would have been obvious to a person of ordinary skill in the art at the time of the invention to incorporate a scanner process to Beier teaching in order to improve data availability.**

With regard to claim 32, the Examiner stated that Beier teaches all limitations as recited except selected scanners to be constraint; however, Beier includes using direct pointers result in a multi-step reorganization process and a prefix update utility (col. 3, lines 37-46 and col. 15, lines 37-50).

With regard to claim 33, the Examiner stated that the database table comprises overflow

pointer records and the original position of a moved record from which a temporary pointer points (abstract).

With regard to claim 34, the Examiner stated the same rejection as provided above for claim 1.

With regard to claim 38, the Examiner stated the same rejection as provided above for claim 18.

With regard to claim 46, the Examiner stated the same rejection as provided above for claim 32.

With regard to claim 47, the Examiner stated the same rejection as provided above for claim 1.

With regard to claim 51, the Examiner stated the same rejection as provided above for claim 18.

With regard to claim 59, the Examiner referred to the rejection as provided above for claims 32 and 33.

A description of the prior art is provided above in Section A.

For the reasons presented below, Appellant respectfully requests that the Board reverse the Examiner's final rejection of Claims 1, 18, 32-34, 38, 46, 47, 51, and 59.

## 2. Claims 1 and 18 are not suggested by Beier

Unpatentability under obviousness requires that the subject matter of the invention would have been obvious to one of ordinary skill in the pertaining art at the time the invention was made.

Claims 1, 18, 32-34, 38, 46, 47, 51, and 59 form a separate and independent group from the group of claims 30, 31, 44, 45, 57, and 58 because these different groups of claims have been

rejected with different rejections by the Examiner under different sections of 37 U.S.C., and accordingly are of different scope and separately patentable.

i. Beier Does Not Suggest Appellant's Invention of Claim 1

Independent claim 1 recites a method for reorganizing a database table online. In particular, claim 1 recites, in pertinent part, the steps of:

1. A method for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the method steps comprising:
  - a) moving a subset of records within the database table;
  - b) flagging each moved record as a reorganization record;
  - c) creating a reorganization pointer record for each moved record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and
  - d) establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization.

Claim 1 is patentable over Beier since the features of claim 1 are not disclosed or suggested by this reference. In particular, Beier fails to teach or suggest the recited features of:

1) moving a subset of records within a database table; 2) flagging each moved record as a reorganization record; 3) creating a reorganization pointer record for each moved record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and 4) establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization.

With regard to the first feature stated above, nothing in Beier teaches, explicitly or

implicitly, or suggests moving a subset of records within a database table. The Examiner stated that Beier teaches the moving of a subset of records within the database table at col. 5, lines 1-6.

However, as explained above with reference to claim 30, Beier only teaches moving data elements from one location to a new location at the cited lines, and does not teach or suggest moving data elements within a stored database table; rather, Beier's invention describes moving data elements from one data set to a completely new data set. Beier therefore does not disclose or suggest the recited element.

With regard to Appellant's second feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests flagging each moved record as a reorganization record. The Examiner stated that Beier teaches flagging each moved record as a reorganization record at col. 6, lines 55-60. These cited lines of Beier describe a "given segment" pointing to a "target segment," where the given segment has a context and reorganization number that indicates the reorganization level of the target at the time the direct pointer is assigned. Beier compares a pointer's reorganization number with a reorganization number assigned to the entire data partition to check whether the pointer is valid or out-of-date (col. 6, line 63 to col. 7, line 8). Beier's reorganization number thus is not a flag for a record that is moved during reorganization, but is a type of timestamp that indicates in which previous reorganization that the pointer to the target segment was updated. In contrast, claim 1 recites that a record that is moved during the current reorganization is flagged as a reorganization record. Beier's reorganization number is not a flag indicating whether a record is moved during a current reorganization, but rather indicates when a direct pointer to a segment was last updated in the past, and thus is not similar to Appellant's flagging step.

With regard to Appellant's third feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests creating a reorganization pointer record for each moved record at the



initial location of the moved record, where the reorganization pointer record points to the new location of the moved record. The Examiner stated that Beier teaches in the abstract “creating” a reorganization pointer record for each moved record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record. However, this is not believed to be the case. Beier’s abstract describes a process in which, for given data elements that point to a target data element, a direct pointer to the target data element is assigned at the time a given data element is created (abstract, lines 13-19). That direct pointer is used to reference the target element unless the pointer is found to be out of date, in which case an indirect pointer is used and the direct pointer is updated. In contrast, Appellant’s claim 1 recites that a reorganization pointer record is created for each record moved during the reorganization and is stored at the initial location of the moved record. Appellant creates a pointer record at the time of reorganization for a record moved during reorganization, not at data element creation as in Beier. Appellant’s pointer record allows old table scanners scanning during a reorganization to scan the database table, find the reorganization pointer record in the table where the moved record used to be, and follow the pointer and find the moved record accurately, as explained in Appellant’s specification on page 20, lines 14-23, for example. In contrast, Beier’s direct pointer is a pointer between existing data elements that is always present with the data elements, is not created during a reorganization, and is updated if it is found to be out of date. Beier’s described steps are used to save reorganization steps so that a separate step of updating all the direct pointers is not needed (see abstract, lines 5-11), and do not themselves allow accurate scans of a database table during a reorganization as does Appellant’s reorganization pointer record of claim 1.

Furthermore, claim 1 recites that a reorganization pointer record is created for each moved record at the initial location of the moved record. In contrast, Beier installs a pointer, not a pointer

record, to the new record location in a locator file (“Indirect List Entry (ILE)”) (col. 5, line 65 to col. 6, line 4) and uses a reorganization number to select between the old and new pointers in the locator file (col. 14, lines 17-38). There is no disclosure or suggestion in Beier’s abstract or elsewhere in Beier to create a reorganization pointer record in the database table at the initial location of the moved record.

With regard to Appellant’s fourth feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization. The Examiner stated that Beier does not explicitly disclose this feature, but the Examiner also stated that it would have been obvious to a person of ordinary skill in the art to incorporate a scanner process in Beier’s teaching in order to improve data availability, citing col. 3, lines 37-46 and col. 15, lines 37-50.

However, similar to the pertinent discussion above for claim 30, the “scan utilities” mentioned by Beier at col. 3 are not used for scans of the database table during reorganization, but are utilities used on data that is not being reorganized. Claim 1, in contrast, recites that the scanner process can retrieve records from the database table during the reorganization of that database table, i.e. data that is being reorganized. The lines at col. 15 of Beier describe the comparing of reorganization numbers, as discussed above, to determine if direct pointers are out of date based on the last update of the pointers during a previous reorganization, and mention nothing about scanner process constraints based on when a scan commences relative to the moving of data during a current reorganization process. Therefore, the cited scanner constraints of claim 1 would not be obvious in view of Beier.

In addition, the multi-step reorganization process that Beier mentions at col. 3, lines 37-40 and col. 4, lines 1-7 is described as keeping the database unavailable for a long period of time, thus

indicating that scanning processes of the database partition are not online and not allowed during the reorganization process of that partition. Beier nowhere mentions how such online scanning would continue from the old data set into Beier's reorganized, new data set, as discussed above for claim 30. Claim 1, in contrast, recites that a scanner process can correctly retrieve records from the database table during reorganization of that table.

Consequently, Beier cannot teach or suggest the method recited in claim 1.

## ii. Beier Does Not Suggest Appellant's Invention of Claim 18

Claim 18 recites, in pertinent part, the steps of:

18. A method for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the method comprising the steps of:

- a) a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers;
- b) a vacate clean up step that removes the temporary pointers of the vacate move step;
- c) a fill move step that moves selected records to the defined set of pages using temporary pointers; and
- d) a fill clean up step that removes the temporary pointers of the fill move step, wherein each clean up step is synchronised to commence at the completion of a move step and to commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

Claim 18 is patentable over Beier since the features of claim 18 are not disclosed or suggested by this reference. In particular, Beier fails to teach or suggest the recited features of: 1) a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers; 2) a vacate clean up step that removes the temporary pointers of the vacate move step; and 3) each clean up step is synchronised to commence at the completion of a move step and to commence only when all queries launching scanner processes

commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

With regard to the first feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers. The Examiner stated that Beier discloses a vacate move step and fill move step at col. 5, lines 1-6; but these lines do not describe the cited move steps, as explained above with reference to claims 1 and 30: Beier only teaches moving data elements from one location to a new location at the cited lines, and does not teach or suggest moving data elements within a stored database table, but rather describes moving data elements from one data set to a completely new data set. Beier therefore does not disclose or suggest the recited element

With regard to the second feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests a vacate clean up step that removes the temporary pointers of the vacate move step. The Examiner stated that Beier discloses a vacate clean up step at col. 1, lines 36-39. However, these cited lines merely describe a general technique of clustering database data to maximize performance, and the loss of clustering due to the deletion and adding of data elements over time during normal database operation (not reorganization). There is no mention of a specific vacate clean up step that removes temporary pointers provided in a vacate move step of a reorganization process as recited in claim 18.

With regard to the third feature stated above, nothing in Beier teaches, explicitly or implicitly, or suggests each clean up step being synchronised to commence at the completion of a move step and to commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion

of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

The Examiner stated that Beier does not explicitly disclose this feature, but the Examiner also stated that Beier discloses using direct pointers result in multi-step reorganization process, a prefix update utility, and that the work files are combined into a sort, so that it would have been obvious to a person of ordinary skill to incorporate a scanner process to Beier teaching in order to improve data availability, citing col. 3, lines 37-46 and col. 15, lines 37-50. However, similar to the pertinent discussion above for claims 1 and 30, the “scan utilities” mentioned by Beier at col. 3 are not used for scans of the database table during reorganization, but are utilities used on data that is not being reorganized. Beier teaches nothing about synchronization of scanner processes with move and clean up steps. The “prefix update utility” mentioned by Beier at col. 3, line 46 refers to a prefix, which is a group of pointers, not data records of a database table. Claim 18, in contrast, recites clean up and move steps synchronized to commence based on queries launching scanner processes of the database table; and the scanner processes can thus occur during the reorganization of the database table. Furthermore, col. 3, lines 37-40 of Beier indicate that the database is unavailable during the multi-step reorganization process (database is only ready to be used after reorganization), such that no queries launching scanner processes are described during reorganization as recited in claim 18. It would not be obvious to include a scanner process during reorganization in Beier, as explained above, e.g., because Beier gives no suggestion how to continue a scanning process from the old data set into his reorganized, new data set.

The lines at col. 15 of Beier describe the comparing of reorganization numbers, as discussed above, to determine if direct pointers are out of date based on the last update of the pointers during a previous reorganization, and mention nothing about the commencement of queries launching scanner processes, or of move steps, relative to the moving, clean-up, and queries of data during a current

reorganization process as recited in claim 18.

Consequently, Beier cannot teach or suggest the method recited in claim 18.

iii. Claims 32-34, 38, 46, 47, 51, and 59

As to claim 32, this claim is dependent on claim 30 and is patentable over Beier for at least the same reasons as claim 30, and for additional reasons. For example, claim 32 recites move steps synchronized with query processes launching scanners. As explained above, Beier does not teach or suggest scanners used during a reorganization process.

As to claim 33, this claim is dependent on claim 30 and is patentable over Beier for at least the same reasons as claim 30, and for additional reasons. For example, claim 33 recites overflow pointer records and the original position of a moved record in the database table, which are not described in the abstract as the Examiner stated; for example, the abstract describes direct pointers which point to a record directly, not to an original position of a moved record.

As to claim 34, Beier does not disclose or suggest online scanning, accessing and updating during reorganization, nor the moving a subset of records, flagging each moved record, creating a reorganization pointer record, and establishing scanner process constraints steps recited therein. Claim 34 is therefore patentable over Beier for reasons similar to claim 1.

As to claim 38, Beier does not disclose or suggest online scanning, accessing and updating during reorganization, nor the vacate move step, vacate clean up step, fill move step, and fill clean up step recited therein. Claim 38 is therefore patentable over Beier for reasons similar to claim 18.

As to claim 46, this claim is dependent on claim 44 and is patentable over Beier for at least the same reasons as claim 44, and for additional reasons. For example, claim 46 recites move steps synchronized with query processes launching scanners. As explained above, Beier does not teach or suggest scanners used during a reorganization process.

As to claim 47, Beier does not disclose or suggest online scanning, accessing and updating during reorganization, nor the means for moving a subset of records, means for flagging each moved record, means for creating a reorganization pointer record, and means for establishing scanner process constraints steps recited therein. Claim 47 is therefore patentable over Beier for reasons similar to claims 1 and 34.

As to claim 51, Beier does not disclose or suggest online scanning, accessing and updating during reorganization, nor the means for performing the vacate move step, vacate clean up step, fill move step, and fill clean up step recited therein. Claim 51 is therefore patentable over Beier for reasons similar to claims 18 and 38.

As to claim 59, this claim is dependent on claim 57 and is patentable over Beier for at least the same reasons as claim 57, and for additional reasons.

Consequently, Beier cannot teach or suggest the subject matter recited in claims 32, 33, 38, 46, 47, 51, and 59.

Accordingly, Appellant respectfully requests that the Board reverse the final rejection of Claims 1, 18, 32-34, 38, 46, 47, 51, and 59.

### **Conclusion**

Because the Beier reference fails to teach or suggest online reorganization allowing a database table to be accessed, scanned, and updated during the reorganization, and the move, vacate and fill steps, and the flagging a moved record, creating a reorganization pointer, and scanner process constraints as recited in claims 1, 18, 30, 34, 38, 44, 47, 51, and 57, the claims are allowable over Beier. Moreover, because claims 31-33, 45, 46, 58 and 59 depend upon their respective parent claims, they are also patentable over Beier.

Accordingly, Appellant respectfully asks the Board to reverse the Examiner's final

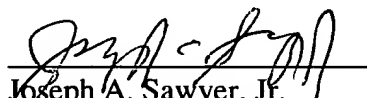
rejection of claims 1, 18, 30-34, 38, 44-47, 51, and 57-59 of the present invention.

**Note: For convenience of detachment without disturbing the integrity of the remainder of pages of this Appeal Brief, Appellant's APPENDIX A is attached on separate sheets following the signatory portion of this Appeal Brief.**

This Brief is being submitted in triplicate, and authorization for payment of the required Brief fee is contained in the transmittal letter for this Brief. Please charge any fee that may be necessary for the continued pendency of this application to Deposit Account No. 09-0460 (IBM Corporation).

Respectfully submitted,

July 27, 2004

  
Joseph A. Sawyer, Jr.  
Attorney for Appellants  
Reg. No. 30,801  
(650) 493-4540





## IX. APPENDIX A

1. (previously amended) A method for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the method steps comprising:

- a) moving a subset of records within the database table;
- b) flagging each moved record as a reorganization record;
- c) creating a reorganization pointer record for each moved record at the initial

location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and

- d) establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization.

2. (original) The method of claim 1 wherein the establishing further comprises:

- d1) identifying table and index scanner processes commencing prior to the completion of the moving as old table scanners wherein old table scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and wherein old table scanners are constrained to ignore reorganization records when accessing records sequentially in a scan of the database table; and

- d2) identifying table and index scanner processes commencing after the completion of the moving as new table scanners, wherein new table scanners are constrained to ignore reorganization pointers in accessing records in the database table.

3. (original) The method of claim 2 further comprising the steps of:

e) deleting the reorganization pointer records and removing the reorganization record flag from the records flagged after each query associated with an old table scanner is complete; and

f) repeating steps a-d after the completion of each query associated with a new scanner until the records in the database table meet a reorganization criteria.

4. (original) The method of claim 3 in which step f) alternates between moving records from a one of successively defined portions of the table and moving records in a predefined order into a one of the successively defined vacated portions of the table.

5. (original) The method of claim 4 in which the predefined order for moving records is defined by a clustering index associated with the table.

6. (original) The method of claim 4 in which the table further comprises overflow pointer records pointing to overflow records and in which the method steps further comprise steps defining each overflow pointer record in a one of the successively defined portions of the table to be a reorganization overflow pointer record, and flagging the associated overflow record to be a reorganization record.

7. (original) The method of claim 4 in which the method steps further comprise the step of preventing the allocation of space for new or updated records in each respective one of the successively defined portions of the table when moving records from the one of successively

defined portions of the table.

8. (original) The method of claim 3 wherein the deleting further comprises requesting and waiting for the exclusive availability of a latch which is shareable by each old table scanner process and which each old table scanner process obtains before commencing scanning and releases after the query associated with the old table scanner process has completed processing.

9. (original) The method of claim 8 in which the latch is shareable by each index scanner process and each index scanner process obtains the latch before commencing scanning and releases the latch after the query associated with the index scanner process has completed processing.

10. (original) The method of claim 3 in which the method steps further comprise steps to maintain the identity and status of moved records in a record identifier mapping table.

11. (original) The method of claim 10 in which the record identifier mapping table comprises a mapping record corresponding to each of the moved records, each mapping record comprising an old record identifier for the initial position of the moved record, a new record identifier for the moved position of the moved record, and a status flag for the moved record.

12. (original) The method of claim 11 in which the status flag for the moved record

comprises flags to indicate the completion of the move of the record, the completion of the clean up for the record, and whether the record has been deleted.

13. (original) The method of claim 10 in which the record identifier mapping table comprises a hash table implemented having a hash on the new record identifier, and a hash table containing a reverse record identifier mapping.

14. (original) The method of claim 3 in which the method steps further comprise logging steps for retaining data values relating to:

- 1) the start of step a),
  - 2) the move of a normal record,
  - 3) the move of a pointer record where, when moving records from a one of successively defined portions of the table, the associated overflow record is not in the vacate range,
  - 4) the move of a pointer record where, when moving records from a one of successively defined portions of the table, the associated overflow record is in the vacate range,
  - 5) the move of an overflow record during a vacate phase,
  - 6) the successful move of a record,
  - 7) the successful clean up of a row that has been moved, and
  - 8) the completion of the online reorganization,
- wherein the table may be rolled back without the loss of data due to the

reorganization of the table.

15. (original) The method of claim 3 in which the method steps further comprise the step of imposing a key value lock where an index scanner is accessing an ordered list of key values in an index associated with the table, the key value lock preventing index values from being updated while the index scanner accesses the ordered list of key values.

16. (original) The method of claim 3 in which the method steps further comprise the step of synchronising between scanners by imposing a share lock on records subject to the moving and records subject to the deleting.

17. (original) The method of claim 3 in which the method steps further comprise an idle step in which neither the moving nor the deleting is performed.

18. (previously amended) A method for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the method comprising the steps of:

- a) a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers;
- b) a vacate clean up step that removes the temporary pointers of the vacate move step;
- c) a fill move step that moves selected records to the defined set of pages using temporary pointers; and
- d) a fill clean up step that removes the temporary pointers of the fill move step, wherein each clean up step is synchronised to commence at the completion of a move step and to

commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

19. (original) The method of claim 18 wherein step a) comprises:

- i) moving each data record in a defined set of pages to a new location in a page in the database table outside the defined set of pages;
- ii) flagging each moved non-overflow data record as a reorganization record;
- iii) creating a reorganization pointer record for each moved non-overflow data record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record;
- iv) for each overflow pointer record located in the defined set of pages and associated with a moved overflow data record, creating a reorganization pointer record for the moved overflow data record at the location of the said reorganization pointer record and flagging the moved overflow record as a reorganization record; and
- v) flagging each overflow pointer record pointing to an overflow record outside the defined set of pages to be a reorganization pointer record, and flagging each of said overflow records as a reorganization record.

20. (original) The method of claim 19 wherein step b) comprises:

- i) deleting reorganization pointers in the defined set of pages; and
- ii) removing the reorganization record flag from each reorganization record in the

database table.

21. (original) The method of claim 20 wherein step c) comprises:

- i) moving selected records to the defined set of pages such that the relative ordering of the selected records matches the predefined ordering constraint;
- ii) flagging each moved record as a reorganization record;
- iii) creating a reorganization pointer record for each moved non-overflow record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record;
- iv) creating a reorganization pointer record for each moved overflow record at the location of the overflow pointer record for the moved record, the reorganization pointer record pointing to the new location of the moved record.

22. (original) The method of claim 21 wherein step d) comprises:

- i) deleting reorganization pointers in the defined set of pages; and
- ii) removing the reorganization record flag from each reorganization record in the database table.

23. (original) The method of claim 22, wherein table scanner processes commencing during each move step are identified as old table scanners, wherein old table scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and are further constrained to access reorganization records sequentially by ignoring reorganization records and table scanner processes commencing after the completion of a move

step are identified as new table scanners, wherein new table scanners ignore reorganization pointers in accessing records in the database table.

24. (original) The method of claim 22 in which each clean up synchronisation step comprises requesting and waiting for the exclusive availability of a latch which is shareable by each old table scanner process and which each old table scanner process obtains before commencing scanning.

25. (original) The method of claim 22 in which the method further comprises the step of preventing the allocation of space for new or updated records in each respective one of a sequentially advancing defined set of pages during the vacate move step for the said one of the sequentially advancing defined set of pages.

26. (original) The method of claim 22 in which an identity and status of moved records is maintained in a record identifier mapping table.

27. (original) The method of claim 26 in which the record identifier mapping table comprises a mapping record corresponding to each of the moved records, each mapping record comprising an old record identifier for the initial position of the moved record, a new record identifier for the moved position of the moved record, and a status flag for the moved record.

28. (original) The method of claim 27 in which the record identifier mapping table comprises a hash bucket comprising a hash on the new record identifier, and a hash table



containing a reverse record identifier mapping.

29. (original) The method of claim 22 in which the database table has an associated index and in which each fill step selects records by scanning the associated index of the database table.

30. (previously amended) A method for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the method comprising the following steps:

- a) a vacate move step to move data records from move pages in the table to available space in the database table; and
- b) a fill move step to move data records into move pages in the table,  
wherein the database table can be accessed, scanned, and updated during the vacate move step and fill move step of the reorganization.

31. (original) The method of claim 30 wherein each move step comprises the step of defining temporary pointers from the original position of each moved record to the moved position of the moved record.

32. (original) The method of claim 31 further comprising the step of

- c) defining selected scanners to be constrained to follow the temporary pointers while accessing records so as to maintain data ordering for the selected scanners wherein the move steps are synchronised with query processes launching scanners so as to maintain the temporary pointers for use by said scanners launched by query processes.

33. (original) The method of claim 32 in which the database table comprises overflow pointer records and the original position of a moved record from which a temporary pointer points.

34. (previously amended) A computer readable medium containing program instructions for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the program instructions comprising the steps of:

- a) moving a subset of records within the database table;
- b) flagging each moved record in as a reorganization record;
- c) creating a reorganization pointer record for each moved record in at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and
- d) establishing scanner process constraints based on whether a scanner process is commenced prior to or after the moving, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization.

35. (original) The computer readable medium of claim 34 wherein step d) further comprises:

- d1) identifying table and index scanner processes commencing prior to the completion of the moving as old table scanners wherein old table scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and wherein old table scanners are constrained to ignore reorganization records when accessing records sequentially in a scan of the database table;

and

d2) identifying table and index scanner processes commencing after the completion of the moving as new table scanners, wherein new table scanners are constrained to ignore reorganization pointers in accessing records in the database table.

36. (original) The computer readable medium of claim 35 further comprising the steps of:

e) deleting the reorganization pointer records and removing the reorganization record flag from the records flagged after each query associated with an old table scanner is complete; and

f) repeating steps a-d after the completion of each query associated with a new scanner until the records in the database table meet a reorganization criteria.

37. (original) The computer readable medium of claim 36 in which step f) alternates between moving records from a one of successively defined portions of the table and moving records in a predefined order into a one of the successively defined vacated portions of the table.

38. (previously amended) A computer readable medium containing program instructions for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the program instructions comprising the steps of:

a) a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers;

b) a vacate clean up step that removes the temporary pointers of the vacate move step;

- c) a fill move step that moves selected records to the defined set of pages using temporary pointers; and
- d) a fill clean up step that removes the temporary pointers of the fill move step, wherein each clean up step is synchronised to commence at the completion of a move step and to commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

39. (original) The computer readable medium of claim 38 wherein step a) comprises:

- ii) moving each data record in a defined set of pages to a new location in a page in the database table outside the defined set of pages;
- ii) flagging each moved non-overflow data record as a reorganization record;
- iii) creating a reorganization pointer record for each moved non-overflow data record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record;
- iv) for each overflow pointer record located in the defined set of pages and associated with a moved overflow data record, creating a reorganization pointer record for the moved overflow data record at the location of the said reorganization pointer record and flagging the moved overflow record as a reorganization record; and
- v) flagging each overflow pointer record pointing to an overflow record outside the defined set of pages to be a reorganization pointer record, and flagging each of said overflow records as a reorganization record.

40. (original) The computer readable medium of claim 39 wherein step b) comprises:

- i) deleting reorganization pointers in the defined set of pages; and
- ii) removing the reorganization record flag from each reorganization record in the database table.

41. (original) The computer readable medium of claim 40 wherein step c) comprises:

- i) moving selected records to the defined set of pages such that the relative ordering of the selected records matches the predefined ordering constraint;
- ii) flagging each moved record as a reorganization record;
- iii) creating a reorganization pointer record for each moved non-overflow record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and
- iv) creating a reorganization pointer record for each moved overflow record at the location of the overflow pointer record for the moved record, the reorganization pointer record pointing to the new location of the moved record.

42. (original) The computer readable medium of claim 41 wherein step d) comprises:

- i) deleting reorganization pointers in the defined set of pages; and
- ii) removing the reorganization record flag from each reorganization record in the database table.

43. (original) The computer readable medium of claim 42, wherein table scanner processes commencing during each move step are identified as old table scanners, wherein old table

scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and are further constrained to access reorganization records sequentially by ignoring reorganization records and table scanner processes commencing after the completion of a move step are identified as new table scanners, wherein new table scanners ignore reorganization pointers in accessing records in the database table.

44. (previously amended) A computer readable medium including program instructions implementing steps for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the steps comprising:

- a) a vacate move step to move data records from move pages in the table to available space in the database table; and
  - b) a fill move step to move data records into move pages in the table,
- wherein the database table can be accessed, scanned, and updated during the vacate move step and fill move step of the reorganization.

45. (original) The computer readable medium of claim 44 wherein each move step comprises the step of defining temporary pointers from the original position of each moved record to the moved position of the moved record.

46. (original) The computer readable medium of claim 45 further comprising the step of

- c) defining selected scanners to be constrained to follow the temporary pointers while accessing records so as to maintain data ordering for the selected scanners wherein the move steps are synchronised with query processes launching scanners so as to maintain the

temporary pointers for use by said scanners launched by query processes.

47. (previously amended) A computer system for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the computer system comprising

means for moving a subset of records within the database table;

means for flagging each record moved by the means for moving a subset of records as a reorganization record;

means for creating a reorganization pointer record for each record moved by the means for moving a subset of records at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and

means for establishing scanner process constraints based on whether a scanner process is commenced prior to or after moving the subset of records, wherein the scanner process can correctly retrieve records from the database table during the reorganization of the database table, including before and after movement of records in the reorganization.

48. (original) The computer system of claim 47 wherein the means for establishing scanner process constraints further comprises:

means for identifying table and index scanner processes commencing prior to moving the subset of records as old table scanners wherein old table scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and wherein old table scanners are constrained to ignore reorganization records when accessing records sequentially in a scan of the database table;

and

means for identifying table and index scanner processes commencing after the completion of moving the subset of records as new table scanners, wherein new table scanners are constrained to ignore reorganization pointers in accessing records in the database table.

49. (original) The computer system of claim 48 further comprising:

means for deleting the reorganization pointer records and removing the reorganization record flag from the records by the means for flagging after each query associated with an old table scanner is complete; and

means for moving another subset of records after the completion of each query associated with a new scanner until the records in the database table meet a reorganization criteria.

50. (original) The computer system of claim 49 in which the means for moving another subset of records alternates between moving records from a one of successively defined portions of the table and moving records in a predefined order into a one of the successively defined vacated portions of the table.

51. (previously amended) A computer system for reorganizing a database table online , allowing the database table to be scanned, accessed and updated during the reorganization, the computer system comprising:

means for performing a vacate move step that relocates a subset of records in a defined set of pages to available space within the database table using temporary pointers;



means for performing a vacate clean up step that removes the temporary pointers of the vacate move step;

means for performing a fill move step that moves selected records to the defined set of pages using temporary pointers; and

means for performing a fill clean up step that removes the temporary pointers of the fill move step, wherein each clean up step is synchronised to commence at the completion of a move step and to commence only when all queries launching scanner processes commenced before the completion of a move step have completed and each move step is synchronised to commence at the completion of a clean up step and to commence only when all queries launching scanner processes after the completion of a previous move step have completed.

52. (original) The computer system of claim 51 wherein the means for performing the vacate move step comprises:

means for moving each data record in a defined set of pages to a new location in a page in the database table outside the defined set of pages;

means for flagging each moved non-overflow data record as a reorganization record;

means for creating a reorganization pointer record for each moved non-overflow data record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record;

means for each overflow pointer record located in the defined set of pages and associated with a moved overflow data record, creating a reorganization pointer record for the moved overflow data record at the location of the said reorganization pointer record and flagging the moved overflow record as a reorganization record; and

means for flagging each overflow pointer record pointing to an overflow record outside the defined set of pages to be a reorganization pointer record, and flagging each of said overflow records as a reorganization record.

53. (original) The computer system of claim 52 wherein the means for performing the vacate clean up step comprises:

means for deleting reorganization pointers in the defined set of pages; and  
means for removing the reorganization record flag from each reorganization record in the database table.

54. (original) The computer system of claim 53 wherein the means for performing the fill move step comprises:

means for moving selected records to the defined set of pages such that the relative ordering of the selected records matches the predefined ordering constraint;

means for flagging each moved record as a reorganization record;

means for creating a reorganization pointer record for each moved non-overflow record at the initial location of the moved record, the reorganization pointer record pointing to the new location of the moved record; and

means creating a reorganization pointer record for each moved overflow record at the location of the overflow pointer record for the moved record, the reorganization pointer record pointing to the new location of the moved record.

55. (original) The computer system of claim 54 wherein the means for performing the fill

clean up step comprises:

- means for deleting reorganization pointers in the defined set of pages; and
- means for removing the reorganization record flag from each reorganization record in the database table.

56. (original) The computer system of claim 55 further comprising:

- means for identifying table scanner processes commencing during the performance of each move step as old table scanners, wherein old table scanners are constrained to access records in the database table by following reorganization pointers to access reorganization records and are further constrained to access reorganization records sequentially by ignoring reorganization records; and

- means for identifying table scanner processes commencing after the completion of the performance of a move step as new table scanners, wherein new table scanners ignore reorganization pointers in accessing records in the database table.

57. (previously amended) A computer system for reorganizing a database table online, allowing the database table to be scanned, accessed and updated during the reorganization, the computer system comprising:

- a) means for carrying out a vacate move step to move data records from move pages in the table to available space in the database table; and

- b) means for carrying out a fill move step to move data records into the move pages in the table,

- wherein the database table can be accessed, scanned, and updated during the vacate move

step and fill move step of the reorganization.

58. (original) The computer system of claim 57 further comprising:

means for defining temporary pointers from an original position of each moved record to the moved position of the moved record;

means for defining selected scanners to be constrained to follow the temporary pointers while accessing records so as to maintain data ordering for the selected scanners;  
and

means for synchronising the move steps with query processes launching scanners so as to maintain the temporary pointers for use by said scanners launched by query processes.

59. (original) The computer system of claim 58 in which the database table comprises overflow pointer records and the computer system further comprises means for setting the original position of a moved record to be the position of an overflow pointer record, from which a temporary pointer points, where the moved data record is originally pointed to by said overflow pointer record.